NBS
PUBLICATIONS

NAT'L INST. OF STAND & TECH R.I.C.
A11105 354551

NBSIR 80-2138

Multiprecision Computing at NBS: Yesterday, Today, and Tomorrow

QC
100
.U56
80-2138
1980
c.2

NATIONAL BUREAU
OF STANDARDS
LIBRARY
SEP 16 1981
not circ
QC100
.U56
no. 80-2138
1980
c.2

NBSIR 80-2138

MULTIPRECISION COMPUTING AT NBS: YESTERDAY, TODAY, AND TOMORROW

Daniel W. Lozier

Mathematical Analysis Division
Center for Applied Mathematics
National Bureau of Standards
U.S. Department of Commerce
Washington, D.C. 20234

October 1980

U.S. DEPARTMENT OF COMMERCE, Philip M. Klutznick, *Secretary*

Luther H. Hodges, Jr., *Deputy Secretary*
Jordan J. Baruch, *Assistant Secretary for Productivity, Technology, and Innovation*

NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

TABLE OF CONTENTS

I. Introduction.....1

II. The Four Basic Arithmetic Operations in Multiprecision.....5

III. Some Representative Examples of Software for Multiprecision Computation.....9

IV. Use of Brent's MP Package with Crary's Augment Precompiler on the NBS Univac 1108 Interactive System.....19

V. Summary and Conclusions.....26

References.....28

Appendix A

Appendix B

Appendix C

ABSTRACT

Multiprecision computing is a technique by which arithmetic operations may be performed on a computer to precision levels that are higher than the directly supported single and double precisions. The last ten years have seen the development of portable Fortran software of very high quality that essentially duplicates all the capabilities of standard Fortran, so that an existing standard Fortran program can be re-executed to arbitrarily high precision. In this paper some of the design techniques for such software, which have evolved at NBS and elsewhere, will be discussed. Methods for using the software presently available at NBS will be described, and a complete example will be given. Directions for further extensions and improvements will be indicated.

I. INTRODUCTION

When setting out to perform a calculation on a computer, a person is faced with a very limited set of possible arithmetic capabilities. One has to choose between single precision, which represents numbers on various computers to a significance in the approximate range from 6 to 15 figures, or double precision, which gives up to approximately 30 significant figures. Even on computers with quadruple precision, about 35 significant figures is the upper limit. The situation with regard to the range of numbers representable in a computer is similar. It varies from approximately 75 orders of magnitude up to several thousand orders of magnitude (nearly 10000 in the new VAX-11 architecture of the Digital Equipment Corporation). Much intellectual and engineering effort has gone into optimizing, according to greatly different criteria by different groups and organizations, the final choice of what arithmetic capabilities will be implemented in hardware. Still, the result is a limited set of possibilities that does not provide the means to do every calculation that we would like to do. Perhaps the people who have a genuine need for more extensive arithmetic capabilities should make a greater effort to influence the computer designers.

If the hardware, or more correctly the computer environment presented to the user, fails to provide for a computational need then our recourse is to programs. This is true, in particular, if the arithmetic capabilities provided by the computer environment are not sufficient for a particular computational task. The relative simplicity of algorithms for the four basic multiprecision operations of addition, subtraction, multiplication and division has undoubtedly led to many undocumented programs as well as to programs that were constructed for use in specific applications. Our interest here is in software, which is distinguished for our purposes from the more general

term "programs" by the criteria of reliability, robustness, correctness, portability, maintainability, availability of documentation, ease of use, completeness, and other desirable qualities. Completeness, in particular, is important since a moment's reflection brings to mind the necessity of having available all of the peripheral functions, such as input-output, type conversion, the mathematical functions, and so on. This leads to a software package of considerable size, and not one that the casual programmer with only a limited commitment is likely to produce.

Fortunately, software possessing most of the desirable properties listed above has been developed during the past decade. A later section of this report will describe some representative contributions that have been made to this development. The MP package of R. P. Brent [1] emerges as the most advanced product.

As in the case of other packages, every arithmetic or other operation performed by Brent's package requires a subroutine call (the package is written in FORTRAN and is intended for use only in the FORTRAN environment). Thus, the function

$$F(x) = \sqrt{2}\, x + \cos x^2 - e^x$$

could be evaluated by the FORTRAN statement

```
F = SQRT(2.0)*X + COS(X**2) - EXP(X)
```

whereas the code

```
CALL  MPCRM(2.0, TWO)        Convert 2 to MP form
CALL  MPSQRT(TWO,TEMP)       Compute √2
CALL  MPMUL(TEMP,X,F)          "    √2 x
CALL  MPMUL(X,X,TEMP)          "     x**2
CALL  MPCOS(TEMP,TEMP)         "     cos x**2
CALL  MPADD(F,TEMP,F)          "    √2 x + cos x**2
```

```
CALL  MPEXP(X,TEMP)          Compute  e^X
CALL  MPSUB(F,TEMP,F)            "     F(x)
```

or something similar would be required to use Brent's package. Here TWO, TEMP, X and F are the names of FORTRAN arrays in which multiprecision numbers are stored. It was to avoid such lists of operations, reminiscent of assembly language code, that FORTRAN was developed in the early days of computers. But FORTRAN was not designed to process MP variables*, which are not accommodated by any standard data type of FORTRAN.

The solution that has evolved for this problem is to extend FORTRAN in such a way as to introduce a new data type for MP numbers. This involves writing a FORTRAN-like compiler, an activity of a completely different kind than writing the MP package itself. Two such compilers have emerged: the FORPAK compiler of D. J. Orser [9] and the AUGMENT compiler of F. D. Crary [4]. Both are written in FORTRAN and produce FORTRAN object code, which is subsequently processed by an ordinary FORTRAN compiler. For this reason the special compilers are known as precompilers. In each case the precompiler is informed about all necessary details of the implementing package for the special computer arithmetic by means of an input "description deck." Thus, they can be used to facilitate the use of any special arithmetic for which a valid description deck can be written.

The precompilers are capable of "parsing" general FORTRAN-like expressions. For example, precompilation of the type declaration

```
MULTIPLE TWO,X,F
```

and the statements

```
TWO = 2.0
F = SQRT(TWO)*X + COS(X**2) - EXP(X)
```

*The representation of MP numbers will be discussed in the next section of this report.

would result in automatic generation of the temporary variable and 8 lines of code given above. One of the precompilers, AUGMENT, is being used widely with Brent's MP package. An example of the use of Crary's precompiler with Brent's package on the NBS central computer will be given in this report. The use of Orser's precompiler with another multiprecision package, the Super Precision Package of Wyatt [9], will be discussed briefly.

II. THE FOUR BASIC ARITHMETIC OPERATIONS IN MULTIPRECISION

We have already remarked that the basic algorithms for addition, subtraction, multiplication and division are simple. They are straightforward implementations of the elementary arithmetic methods taught by the school system, easily done in FORTRAN using integer arithmetic.

Many details on multiprecision arithmetic are given in Knuth [6] and references listed there. We give only a brief overview here. The most common approach, and the one used in Brent's MP package, is the following. Some number of consecutive integer storage locations (or words), say n + 2, is decided upon for the representation of MP numbers. A real number to be represented in MP form is first expressed in a notation like scientific notation:

$$a = \pm (0.a_1a_2a_3 \ldots)b^e.$$

Here $b > 0$ is an integer (the base or radix) whose optimal value depends on the computer wordlength, and the a_i are the (unique) digits of a to the base b. That is, $0 \leq a_i < b$ for each i. The representation is made unique by requiring the first digit to be nonzero so that we have $1 \leq a_1 < b$. The representation is then said to be normalized; $0.a_1a_2a_3 \ldots$ is called the (normalized) fraction of a and e is called the exponent of a. The MP form of a is stored by placing the sign (equal to -1 or +1) in word 1, the exponent in word 2, and the first n digits of the normalized fraction in words 3 through n + 2. The final digit may be modified according to some rounding rule. The representation of zero in MP form is nonunique; word 1 is set equal to zero and words 2 through n + 2 are undetermined.

Now it becomes clear how the basic arithmetic operations can be coded using FORTRAN integer arithmetic. After taking into account any necessary shifting of the fractions, the sum of two fractions is obtained by ordinary

digit-by-digit "add and carry" operations. The difference of two fractions is obtained by ordinary digit-by-digit "borrow and subtract" operations. These operations are O(n) in number and this order cannot be improved.

The full multiplication of two fractions requires each digit of one fraction to be multiplied by every digit of the other, together with appropriate shifting and scaling operations, in order to produce the 2n-digit product. Some practical savings can be obtained by not computing all of this, since only n digits of the result are to be stored, but this procedure still requires $O(n^2)$ operations. Multiplication algorithms of lower order exist but have not found their way into general purpose multiprecision packages, for the reason that excessively large values of n are required before actual savings in execution time can be achieved [1,9]. The "fast" algorithms require more "overhead" which tends to dominate the multiplication process for moderate values of n. As an indication of the execution times that can be expected, for Brent's package on the UNIVAC 1108 the multiply time with n = 10 is approximately 2 msec and with n = 14 it is approximately 2.8 msec. These values of n correspond approximately to 43 and 62 decimal places of precision. For comparison, the hardware double precision (18 decimals) multiply time is 5μsec.

However, one special multiplication algorithm is often implemented. It is found, for example, in Brent's package. When one of the operands is a FORTRAN integer then multiplication requires only O(n) operations. But it is interesting to note that this capability cannot be fully exploited by the precompiler. If the FORTRAN statement is

```
X = A * I
```

where X, A are multiprecision variables and I is a FORTRAN integer variable, then the desired O(n) multiplication is performed. Alternatively, suppose I

is a multiprecision variable with every digit of its fraction equal to zero after the first digit. Then I is representable as a FORTRAN integer but the precompiler cannot know to use the O(n) multiplication. We conclude that, in this case, writing FORTRAN code with mixed mode expressions is preferable to the common programming practice of matching the types of all elements in a FORTRAN expression.

Division is also an $O(n^2)$ operation, whether it is done by (i) reciprocating the denominator by use of Newton's method followed by multiplication by the numerator or (ii) the school, or divide-and-correct, method. Brent's package uses method (i) but other programs, such as Wyatt's, use method (ii). As in the case of multiplication, "fast" division methods have not been used in general-purpose multiprecision software, except for the special case when the divisor is typed as a FORTRAN integer. This special case has order n and is exploitable by the precompiler only when the FORTRAN source statement mixes multiprecision and integer types of variables.

We conclude this section with remarks on the optimal value of the base b of representation of MP numbers. When two fractions are multiplied, each digit of one is multiplied by every digit of the other. Since the largest value of a digit is b-1, it is clear that $(b-1)^2$ must not overflow a FORTRAN integer storage location. Thus, there is an upper limit on the allowable bit-length of digits equal to approximately a half-wordlength. The actual upper limit for b in Brent's software is specified by the condition that $8b^2-1$ may not exceed the largest machine-representable integer.

For storage reasons we want b to be as large as possible. This suggests that b should be a power of 2. However, if a great deal of input and output is to be done, then a power of 10 might be preferable because of the cost of conversion between binary and decimal representations. The following table,

taken from Brent's documentation [2] of the MP package, shows optimal power-of-2 and power-of-10 choices for b as a function of wordlength:

Wordlength	Power-of-2 Choice	Power-of-10 Choice
48 bits	4194304	1000000
36 bits	65536	100000
32 bits	16384	100000
24 bits	1024	1000
18 bits	128	100
16 bits	64	10
12 bits	16	10

III. SOME REPRESENTATIVE EXAMPLES OF SOFTWARE FOR MULTIPRECISION COMPUTATION

Multiprecision computing can be viewed as having reached a fairly advanced level of development in Brent's package, particularly when used in combination with a precompiler such as Crary's. Other forms of special computer arithmetic, less related to floating-point computation and to the FORTRAN language, have also evolved but will not concern us here. In this section we will describe briefly several representative examples of earlier multiprecision work. Then we will describe Brent's package in somewhat more detail.

Lawson's Q-Precision Package

Development of this package was begun in 1964 or even earlier by C. L. Lawson at Jet Propulsion Laboratories. Originally for the IBM 7094 and later for the UNIVAC 1108, early versions were written mostly in assembly language. These evolved into the present version in which FORTRAN is used extensively, but the core of the program remains in assembly language. The package is limited to triple and quintuple precision with no extension of the exponent range. It includes many elementary mathematical functions, since it was used to test the accuracy of the FORTRAN function library, as well as a Gauss elimination subroutine for solving linear systems. Its limited design goals permitted the use of function approximations that have an intrinsic precision limit associated with them (Chebyshev and minimax approximations). Such approximations are favored for efficiency but are not suitable for a general multiple-precision capability. The FORTRAN that was used contains nonstandard expressions. The Q-precision package suffers badly from lack of portability but was never viewed as general-purpose software. It was never published in

the open literature. Nevertheless, it serves as an excellent early example because of its successful applications on two different computers.

Peavy's Package

Like Lawson, B. A. Peavy of NBS began to develop a multiprecision package in the 1960's. But Peavy's package is coded entirely in FORTRAN. However, it lacks portability because it uses nonstandard FORTRAN expressions and was designed with only the UNIVAC 1108 in mind. It includes addition, subtraction, multiplication and division; square root, sine, cosine, arctangent, exponential and logarithm; double precision to multiple precision conversion and the reverse conversion; comparison and printing capabilities. The package uses from 3 to 20 words per multiprecision variable, the precise number being set by the statement CALL ISTART(N). The package stores 10 decimal digits per word, with the final 5 digits of the last word representing the exponent. Thus precisions from 25 to 195 in steps of 10 are provided, with the exponent (to the base 10) ranging from -4999 to 4999. Changing from one precision to another is not facilitated by this choice of representation, which can be a drawback for the use of the package. The package has been used by I. A. Stegun and R. Zucker of the Center for Applied Mathematics of NBS to test subroutines for the sine, cosine, exponential, hyperbolic cosine integrals, and to provide check values to 35 significant figures for these functions. Unfortunately, no publication describing Peavy's package was ever prepared. An informal write-up does exist, however.

Maximon's Package [8]

The motivation for development of this package, which appeared in 1971, was somewhat different from Lawson's and Peavy's. L. C. Maximon, an NBS

physicist, needed higher precision than was available on the UNIVAC 1108 in order to compute "cross sections for the scattering of high energy electrons and ... protons from nuclei." [8, p2]. The theoretical physicist/applied mathematician's point of view is well stated by Maximon: "Although analytical methods may be developed for a particular problem, the most direct and generally applicable technique is simply to perform the pertinent part of the calculation (that part involving the cancellation with resultant loss of significant figures) with however many significant figures are needed in order that the final result have the desired number of significant figures." [8, pp2,3]. Maximon also recognized two other important applications of multiprecision arithmetic, namely the construction of multiprecision function routines for the purpose of testing single, double or other fixed precision library routines, and the computation of the special functions of mathematical physics to arbitrary precision.

Maximon's package is written entirely in FORTRAN with a view toward portability. It provides for addition, subtraction, multiplication, division, and division by an integer; conversion from integer, single or double precision variables to extended precision and the reverse conversions to double precision; and output printing. The working precision, number of base digits per word, and the base (2,8,10 or 16) are specified by the user by calling an initialization routine, PRM(NDG,NDW,NBRT). Parameters derived from these inputs are communicated to the working subroutines via COMMON. Appropriate values for NDG, NDW and NBRT depend on the wordlength of the computer and the declared DIMENSION lengths of the multiprecision number arrays. The ability to change these values at any time during the execution of the user's program, within the operable limits, is a notable feature. It was absent in Lawson's and Peavy's packages but is present in the packages of Wyatt and Brent. The

change is made by calling the initialization subroutine with the desired new parameter values.

Subroutines to evaluate selected mathematical functions were added later to the package. These were used to test the accuracy of the UNIVAC 1108 Fortran Library [7].

A major design problem for any multiprecision package is how to handle input and output. The problem does not stem from the requirement for base conversion between incommensurate number bases, although accurate and fast algorithms are not completely straightforward when numbers of all sizes are to be accommodated. The problem is how to manage the physical arrangement of numbers to be read in or printed out. In FORTRAN the FORMAT statement exists for this purpose. FORMAT statements require interpretive processing at run-time, a task of greater magnitude in its full generality than any multiprecision package designer has yet undertaken, as far as this author is aware. Consequently, simplified forms are found in all of the multiprecision packages considered here.

Input is relatively simple; some form of free-form input format is the usual solution. This, of course, requires character processing as well as numerical processing. Maximon does not provide for number input except by way of conventional FORTRAN variables which can be converted (in their restricted precision) to multiple-precision form. This was reasonable for the types of applications he had in mind. The output of a single multiprecision number is usually done by encoding a character string which then is printed using a FORTRAN format of the form nA1. This may have to be continued on several lines if the precision is very high. Maximon considered the output problem with some care and came essentially to this solution. Annoyingly, the exponent, although printed in decimal form, refers to the internal number base

which could be other than 10. He provided for grouping the digits to improve readability and for specifying the number of digits to print before the decimal point. Changing these specifications is cumbersome and may require recompilation of one subroutine. The packages of Wyatt and Brent are more flexible in this regard.

Finally, we note that Maximon's package includes no detection of underflow or overflow. The handling of errors in general is another difficult design question which has received much attention from later multiprecision software designers.

Wyatt's Package [9]

W. T. Wyatt, Jr. of Harry Diamond Laboratories is a physicist. Like Maximon, he needed higher precision than was (or is now) available on conventional computers. In an ambitious undertaking he produced portable FORTRAN software for "Super Precison" computing that essentially duplicated all of the capabilities of the FORTRAN language. An unpublished document describing his package is dated March 21, 1973. In addition to the four basic arithmetic operations for Super Precision operands, and Super Precision mixed with integer operands, he provided for complex arithmetic with Super Precision real and imaginary parts. Both the real and the complex mathematical functions found in the 1966 FORTRAN standard language* were included, as were the so-called "intrinsic functions" --MOD, ABS, INT, MIN, MAX, etc. Wyatt's motivation was his conviction that every existing FORTRAN program should be convertible for use with Super Precision arithmetic. This inevitably led to the idea of a precompiler, for the reasons we have already discussed. Without a precompiler,

*See American Standard FORTRAN, X3.9-1966, published by American Standards Association, Inc., now superseded.

every arithmetic expression would have to be "parsed" by a programmer and recoded, operation after operation. A fruitful collaboration with D. J. Orser of the Applied Mathematics Division of NBS (now the Center for Applied Mathematics) resulted in Orser's FORPAK precompiler [9].

Before discussing FORPAK, we describe some of the design features of the Super Precision Package. By means of a call on the initializing subroutine, ZSETUP(I,J,K), the user selects the number of digits I for the representation of multiprecision numbers; the number base J from the set 2,3,...,16; and specifies the number of bits K in the computer integer word. The software determines the minimum number of words, NF, needed to meet or exceed the requested number of digits. The number of digits per word is determined so as to minimize the amount of storage used. Accordingly, Wyatt's package incurs the overhead of unpacking and repacking operations for every multiplication so as to avoid the necessity of wasting more than half of every word. After the call to ZSETUP, computation proceeds using approximately rounded arithmetic, "approximately" meaning that something less than the full double length product of two multiprecision fractions is formed during multiplication.

Wyatt's package permits the user to exercise additional control over its operation by means of global variables in labelled COMMON blocks. For example, alternative rounding strategies can be selected. Another example is "clip mode". In clip mode the excess digits, beyond digit I, are set to zero after every arithmetic operation and rounding takes place in digit I instead of in the final digit of word NF. This gives the user maximum control over the precision. The desirability of this feature for error analysis has been emphasized by T. E. Hull [5].

Wyatt's package includes an error-handling and traceback capability, in contrast to the previous packages considered here. When an error is detected, such as an attempted division by zero or a negative argument for the square root function, the subroutine that detected the error prints a message and calls the traceback subroutine. The traceback subroutine prints the chain of subroutine calls through the Super Precision Package that led to the error. Execution is then terminated, but the user can easily modify the traceback subroutine to perform a "standard fixup and return".

As previously mentioned, Orser's precompiler was developed with Wyatt's package in mind. Nevertheless, it is, like Crary's precompiler, a general precompiler suitable for use with any new data type for FORTRAN which has a suitable supporting software package. Orser and Wyatt had as a goal to minimize the number of changes necessary to transform an existing conventional FORTRAN program into a program making valid use of the new SUPER PRECISION data type. This goal implied that multiprecision variables appearing in input/output statements should be treated automatically. Accordingly, a method was developed to analyze the original FORMAT statements. Let us consider only the more interesting case of output formats. If no Super Precision variable appeared in an output list, the corresponding FORMAT statement was not altered and the output operations proceeded identically in both the original and transformed programs. On the other hand, suppose a Super Precision variable does appear. Then the FORMAT statement is divided into substatements by means of the numeric designators which correspond to Super Precision variables in the transformed program. These substatements are processed like ordinary FORMAT statements (they involve no nonstandard FORTRAN variables), with a special fixed form of numerical output of the Super Precision variables in between the output they generate. Thus, all output headings from the original

program are preserved. Each Super Precision number is printed starting on a new line and continuing onto additional lines as necessary. In contrast to the current arrangement with the Brent-Crary software, no programmer intervention is required.

Details of usage of the Wyatt-Orser software will not be given here. The ease of its usage is comparable to the Brent-Crary software which is described in detail in the next section of this report. Interested individuals are invited to contact either this author or D. J. Orser.

Brent's Package [1]

The design of Brent's package has already been discussed in some detail. It includes most features of Wyatt's package with some exceptions such as "clip mode". It has quite a few more functions and fundamental constants. From a computer programming point of view, it is more cleanly designed and implemented. This has important implications for the future maintenance and modification of the package. Unlike Maximon and Wyatt, Brent is a numerical analyst with strong interest in complexity theory and algorithm construction. This has not resulted, in many cases, in very much different algorithms for arithmetic and function procedures but Brent states his algorithms more explicitly. This also contributes to the maintainability of the MP package.

The MP User's Guide [2] provides complete documentation of Brent's MP package, including its usage with Crary's AUGMENT precompiler. AUGMENT and MP, as well as the MP User's Guide itself, are resident in the NBS UNIVAC 1108 Interactive System, hereafter called System I. Current information for accessing this documentation and for using the software will be given in the next section.

We have already introduced the (unpacked) representation of numbers and the four basic arithmetic operations in MP. In addition to the unpacked representation, there exists a packed representation in which two digits, instead of only one, are stored in each word. The use of the packed representation saves storage at the expense of time.

The range of MP numbers is limited by the integer wordlength, since only one word is used for the range. This is also true of Wyatt's and Maximon's packages. On System I this translates into an exponent range of approximately $10^{\pm 20686623784}$ (i.e., B^M with $B = 2^{16}$, $M = 2^{32}$). Underflow in an arithmetic operation results in the storing of zero; overflow causes termination with an error message. For the arithmetic operations the user has the choice of truncation (the default choice), proper rounding, or directed rounding. The directed roundings (round up and round down) are intended for an interval arithmetic extension of MP that Brent is planning to implement.

Another useful extension of the package would be to allow a variable number of words to store the exponent instead of just one. This extension of range is needed now in order to test Clenshaw and Olver's "unrestricted" exponential algorithm [3]. This algorithm produces values of e^x guaranteed correct to any relative precision α, for any input argument x. The algorithm is self-adaptive, i.e., it selects parameters so that an estimate of the running time is minimized asymptotically for large values of x and/or small values of α. Brent's package suffices to test the algorithm for small values of α but not for sufficiently large values of x. Although Brent acknowledges the need for a multiword range, he presently plans generalizations of the package in other directions. Consequently, it appears to be necessary to undertake the multiword range extension of MP here at NBS.

Appendix A is taken from the MP User's Guide [2]. It gives the names of all MP subroutines that are likely to be called from a user's program. Most of the names are suggestive of their functions. Every name has the prefix MP. The suffix I indicates an arithmetic operation involving a multiprecision operand with a FORTAN integer operand. The suffix Q indicates an arithmetic operation involving a multiprecision operand with a rational operand, represented by two FORTRAN integers (the numerator and denominator). As an example of usage typical of all MP subroutines, CALL MPADDQ (X,I,J,Y) adds the rational number I/J to the multiprecision number X, giving the multiprecision number Y. Full details on the direct usage as well as the preferred usage via AUGMENT of all MP subroutines may be found in the MP User's Guide.

IV. USE OF BRENT'S MP PACKAGE WITH CRARY'S AUGMENT PRECOMPILER ON THE NBS UNIVAC 1108 INTERACTIVE SYSTEM

Files* exist in the NBS Univac 1108 Interactive System, also referred to as System I, which facilitate the use of Brent's MP package with Crary's AUGMENT precompiler. Although many possibilities exist, we restrict ourselves here to consideration of only one type of usage with a choice of two different precisions: 43 and 62 decimal places of precision. First we present the suggested EXEC 8 runstream. Then we discuss typical aspects of preparing the individual program units for precompilation. The detailed user's guide to the use of Brent's package resides in AMDAI*MP.USERSGUIDE. This guide includes pertinent information on the use of AUGMENT as well as the MP package.

The type of usage we consider is the conversion of a user-written FORTRAN program to multiprecision from ordinary FORTRAN precision. The FORTRAN program consists of one or more program units (the main program plus FUNCTION and SUBROUTINE subprograms), let us say MAIN, FUN1, FUN2, SUB1, SUB2. It may be that not every program unit requires multiprecision, and therefore precompilation, but let us assume that they all do. If any do not then they would be handled according to the usual rules of EXEC 8. The following runstream is prepared:

```
1.  @ASG,A      AMDAI*AUGMENT.
2.  @ASG,A      AMDAI*MP.
3.  @XQT        AMDAI*AUGMENT.AUGMENT
4.  @ADD        AMDAI*MP.DESCRIPTION/MP12
```

*It is necessary to assume some familiarity with certain system aspects of UNIVAC 1100 Series computers. No attempt will be made here to define the many terms, such as "files", which have precise technical definitions peculiar to these computers. Reference: UNIVAC Series 1100 Executive System, Vol. 2, EXEC Level 36R2 Programmer Reference, UP-4144.23, 1979, or successor, published by Sperry Rand Corp.

```
 5.  *BEGIN
 6.  :FOR,IS    MAIN
 7.  Source deck for MAIN
 8.  :FOR,IS    FUN1
 9.  Source deck for FUN1
10.  :FOR,IS    FUN2
11.  Source deck for FUN2
12.  :FOR,IS    SUB1
13.  Source deck for SUB1
14.  :FOR,IS    SUB2
15.  Source deck for SUB2
16.  *END
17.  @ADD   20.
18.  @MAP,IS    ,MAIN
19.  IN MAIN, FUN1, FUN2, SUB1, SUB2
20.  IN AMDAI*MP.MPINIT/MP12
21.  LIB AMDAI*MP.
22.  END
23.  @XQT       MAIN
24.  Input data, if any
```

The result of this runstream will be the multiprecision execution of the user's FORTRAN program to the equivalent of 43 significant figures. If 62 significant figures are required, the number 12 in items 4 and 20 must be changed to 16 (the numbers 12 and 16 indicate the number of FORTRAN integer storage locations used to represent each MP number). We remark that (i) at present only the FORTRAN V compiler can be used, which means :FTN cannot be substituted for :FOR in items 6, 8, 10, 12, 14; (ii) the non-EXEC 8 symbol :

is changed to @ during processing by AUGMENT; and (iii) the output of AUGMENT is a legitimate system runstream stored in File 20 and initiated by line 17 of the runstream.

The best way of illustrating the typical aspects of preparing an individual program unit for pre-compilation is by means of an actual example. Figure 1 shows an ordinary EXEC 8 runstream of the most common type: a program is read in, compiled and executed using every available system default. This is an example of the runstream created when a naive user of System I submits a card deck over the counter for processing.

Figure 2 shows the modified runstream ready for precompilation and execution. It is easily verified that all of the EXEC 8 and AUGMENT control cards are in place according to the suggested runstream presented above. We now focus attention on the actual FORTRAN code in Figure 1 and its transformation for multiprecision processing in Figure 2, referring to individual lines of the codes by their line numbers. The transformation process*, which must be carried out by the programmer, is analogous to transforming a single-precision program into double precision. Type declarations, constants, and input/output statements must all be reviewed and modified appropriately.

Figure 1, Line 2 is replaced by Figure 2, Lines 8,9,10. This changes every floating-point variable from double precision to multiple precision, which in this example is the equivalent of 62 decimal places as specified by the number 16 appearing in Figure 2, Lines 4 and 78.

Figure 1, Line 4 is replaced by Figure 2, Line 14. This replaces the double-precision value of the constant π by its 62-place multiprecision

*The procedures to be described here are suitable for use with the 1978 version of Brent's package. In 1980 an updated version appeared in which more powerful input/output subroutines support noticeably simpler procedures. For further information, contact the author of this report.

```
 1:@FOR,IS FORM
 2:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
 3:       DIMENSION F(41),G(41),U(41)
 4:       PI=3.14159265358979323D0
 5:   111 READ(5,666,END=4000)ALPHA,R
 6:       WRITE(6,777)ALPHA,R
 7:   666 FORMAT(2F8.4)
 8:   777 FORMAT(1X,6HALPHA=,F8.4,10X,2HR=,F8.4)
 9:       AR=ALPHA*R
10:       DO 77 N=20,40,10
11:       NM1=N-1
12:       NP1=N+1
13:       FLN=N
14:       D=PI*ALPHA**3*DEXP(-AR)/(2.D0*AR*FLN)
15:       F(1)=1.D0
16:       F(2)=1.D0-2.D0*AR
17:       DO 10 K=1,NM1
18:       FLK=K
19:       F(K+2)=((2.D0*FLK+1.D0-2.D0*AR)*F(K+1)-FLK*F(K))/(FLK+1.D0)
20:    10 CONTINUE
21:       DO 3 KK=1,NP1
22:       WRITE(6,1000)KK,F(KK)
23:     3 CONTINUE
24:       WRITE(6,2000)
25:       G(1)=0.D0
26:       G(2)=-6.D0*AR+2.D0*AR*AR
27:       DO 20 K=2,NM1
28:       G(K+1)=F(K+2)+F(K+1)-F(K)-F(K-1)
29:    20 CONTINUE
30:       DO 5 KK=1,N
31:       WRITE(6,1000)KK,G(KK)
32:     5 CONTINUE
33:       WRITE(6,2000)
34:       W(1)=0.D0
35:       DO 100 I=0,N
36:       FLI=I
37:       B=FLI*PI/FLN
38:       CSB=DCOS(B)
39:       W(2)=2.D0*(1.D0+CSB-AR)
40:       DO 30 K=1,NM1
41:       W(K+2)=2.D0*CSB*W(K+1)-W(K)+G(K+1)
42:    30 CONTINUE
43:       ANI=1.D0
44:       IF(I.EQ.0)ANI=0.5D0
45:       IF(I.EQ.N)ANI=0.5D0
46:       C=ANI*(-1)**I*D
47:       W(N+1)=C*W(N+1)
48:       WRITE(6,1000)I,W(N+1)
49:   100 CONTINUE
50:       WRITE(6,2000)
51:    77 CONTINUE
52:       GO TO 111
53:  1000 FORMAT(1X,I3,5X,D25.18)
54:  2000 FORMAT(1H0)
55:  3000 FORMAT(1X,2D30.18)
56:  4000 CONTINUE
57:       STOP
58:       END
59:@XQT
60:    0.5D0   1.0D0
```

FIGURE 1. A typical UNIVAC 1108 runstream.

```
@ASG,A AMDAI*AUGMENT.
@ASG,A AMDAI*MP.
@XQT AMDAI*AUGMENT.AUGMENT
@ADD AMDAI*MP.DESCRIPTION/MP16
*BEGIN
!FOR,IS FORM16
      DOUBLE PRECISION RALPHA,RR
      MULTIPLE ALPHA,ANI,AR,B,C,CSB,D,F,FLI,FLK,
     1         FLN,G,PI,R,U
      MULTIPLE MPEXP,MPCOS
      DIMENSION F(41),G(41),U(41)
      DIMENSION MPFRA(64)
      INITIALIZE MP
      CALL MPPI(PI)
  111 READ(5,666,END=4000)RALPHA,RR
      ALPHA=RALPHA
      R=RR
      WRITE(6,777)RALPHA,RR
  666 FORMAT(2F8.4)
  777 FORMAT(1X,6HALPHA=,F8.4,10X,2HR=,F8.4)
      AR=ALPHA*R
      DO 77 N=20,40,10
      NM1=N-1
      NP1=N+1
      FLN=N
      D=PI*ALPHA**3*MPEXP(-AR)/(2.D0*AR*FLN)
      F(1)=1.D0
      F(2)=1.D0-2.D0*AR
      DO 10 K=1,NM1
      FLK=K
      F(K+2)=((2.D0*FLK+1.D0-2.D0*AR)*F(K+1)-FLK*F(K))/(FLK+1.D0)
   10 CONTINUE
      DO 3 KK=1,NP1
      CALL MPOUTE(F(KK),MPFRA,MPCHR,64)
      WRITE(6,1000)KK,MPFRA,MPCHR
    3 CONTINUE
      WRITE(6,2000)
      G(1)=0.D0
      G(2)=-6.D0*AR+2.D0*AR*AR
      DO 20 K=2,NM1
      G(K+1)=F(K+2)+F(K+1)-F(K)-F(K-1)
   20 CONTINUE
      DO 5 KK=1,N
      CALL MPOUTE(G(KK),MPFRA,MPCHR,64)
      WRITE(6,1000)KK,MPFRA,MPCHR
    5 CONTINUE
      WRITE(6,2000)
      U(1)=0.D0
      DO 100 I=0,N
      FLI=I
      B=FLI*PI/FLN
      CSB=MPCOS(B)
      U(2)=2.D0*(1.D0+CSB-AR)
      DO 30 K=1,NM1
      U(K+2)=2.D0*CSB*U(K+1)-U(K)+G(K+1)
   30 CONTINUE
      ANI=1.D0
      IF(I.EQ.0)ANI=0.5D0
      IF(I.EQ.N)ANI=0.5D0
      C=ANI*(-1)**I*D
      U(N+1)=C*U(N+1)
      CALL MPOUTE(U(N+1),MPFRA,MPCHR,64)
      WRITE(6,1000)I,MPFRA,MPCHR
  100 CONTINUE
      WRITE(6,2000)
   77 CONTINUE
      GO TO 111
 1000 FORMAT(1X,I3,5X,64A1,'E',I3)
 2000 FORMAT(1H0)
 3000 FORMAT(1X,2D30.18)
 4000 CONTINUE
      STOP
      END
*END
@ADD 20.
@MAP,IS ,FORM16
IN FORM16
IN AMDAI*MP.MPINIT/MP16
LIB AMDAI*MP.
END
@XQT FORM16
   0.5D0   1.0D0
```

FIGURE 2. The runstream modified for multiprecision execution.

value. If this were not done, then errors in the 18th decimal place would enter the multiprecision calculation, thereby invalidating it. The effect is the same when a single-precision constant inadvertently enters a double-precision calculation.

Figure 1, Lines 5,6,7 and 8 read input values and print them out. The values used in the multiprecision calculation should be identical. The transformed code appearing in Figure 2, Line 7, 15-20 accomplishes this. The technique simply introduces new double-precision variable names for the input variables and establishes the multiprecision equivalents by means of the assignment statements in Figure 2, Lines 16 and 17. The input and output conversion is done by System I identically in Figures 1 and 2. The establishment of multiprecision equivalents by assignment is done exactly if the MP package is working in a base that is a power of two (which it is if the techniques presented here are used).

Output of numbers which will be represented in multiple precision is indicated in Figure 1, Lines 22, 31 and 48 according to the format of Line 53. The transformed code appearing in Figure 2, Lines 12,34,35,44,45,62,63,68 accomplishes the corresponding multiprecision output. The technique introduces a new storage vector of length 64, equal to one storage location for each of the 62 decimal digits plus one for the sign and one for the decimal point. See Figure 2, Line 12. The MP subroutine MPOUTE encodes the fraction part of a single multiprecision number by storing these 64 symbols into the storage vector, and the exponent part of the multiprecision number into an integer storage location. Output of the resulting decimal-encoded number is controlled by the transformed FORMAT statement appearing in Figure 2, Line 68. Minor variations on this technique will permit some degree of flexibility, such as the printing of two columns of numbers instead of one. It should be

noted that the role of the number 64 would be played by 45 if we were working to 43 decimal places.

One final modification of the original code must always be done. It is the insertion of the statement INITIALIZE MP as the first executable statement of the transformed program; see Figure 2, Line 13.

The result of processing the transformed runstream of Figure 2 using System I is shown in Appendix B. We see that the printed output of the AUGMENT precompiler consists of the listing of the "description deck" for the MP package (introduced by Figure 2, Line 4) followed by a commented version of the original FORTRAN source. In our example the only comments are indications of occurrences of mixed-mode operations. If there were any errors detected by AUGMENT, they would be identified here by comments. As stated earlier, the legitimate FORTRAN code produced by AUGMENT is stored as part of a system runstream in File 20. This runstream is initiated in Figure 2, Line 75, and it results in the invocation of the FORTRAN V compiler. The compiler output clearly shows the expansion of FORTRAN expressions into lists of subroutine calls on the MP package. Following this is the output listing of the MAP processor and, finally, the multiprecision results of execution of the program.

Appendix C shows the computer-generated accounting information for the original runstream of Figure 1 and the transformed runstream of Figure 2. We observe factors of 7.66 in Total Time, 24.76 in CPU Time, 7.98 in Core Block Seconds, and 2.74 in Total Run Cost.

V. SUMMARY AND CONCLUSIONS

Interest in multiprecision computing has existed since the earliest days of modern computing. This interest has led to many implementations of multiprecision programs. But only recently have truly general-purpose software packages of high quality been developed. We have seen three representative examples of earlier attempts to produce software of high quality, namely the packages of Lawson, Peavy and Maximon. Then we considered the package of Wyatt, important for its thoroughness and completeness as well as its stimulation of the precompiler of Orser. The close connection between compilation processes and multiple-precision computation was joined in actual software. Finally, we considered the package of Brent, the most advanced multiple-precision package known to this author. Not only is it built on a firm theoretical foundation, it also employs disciplined coding techniques that ease the burden of making additions, corrections and modifications. Furthermore, it is distributed with a user's guide, installation instructions, test programs, and an interface which enables the use of the precompiler of Crary, all on the same source magnetic tape. For these reasons and the fact that it is actively supported by its author, the use of Brent's package with Crary's precompiler is to be recommended for most applications of multiple-precision arithmetic. However, the use of Brent's package with Orser's precompiler is an alternative that should be investigated.

It should not be considered, however, that Brent's package leaves nothing to be done. Additional mathematical functions could be added to the present repertory, which includes in addition to the FORTRAN library functions a limited complement of special functions (Bessel's function of the first kind, Dawson's integral, exponential integral, error function, complementary error function, gamma function, logarithmic integral, log gamma function). Improved

algorithms for these functions could be implemented. A more fundamental modification of Brent's package would be to extend the MP number representation so that an arbitrary number of words could be used for the exponent part as well as the fraction part. Presently, only one word is used for this purpose. Such an extension would make it possible to design algorithms in which underflow and overflow can always be avoided. Although exponents of one word are satisfactory for many applications, current research in algorithms at NBS and elsewhere is being hampered right now by this limitation. Finally, we note that Brent's own future development of the MP package seems to be centering on the provision of an interval arithmetic capability.

Another important activity for the future would be to develop micro-coded "instructions" for multiprecision operations on a micro-codable computer. This would speed up multiprecision computations on that computer but the major purpose would be to gain additional knowledge about arithmetic processes in general. For example, one could attempt to build an efficient system for computing with base, precision, and range completely under the control of the programmer. The system would be a prototype for future developments in the design of arithmetic units for general-purpose computers. It also would have an influence on future compiler requirements. The resulting freedom from unnatural restrictions on the precision and range of floating-point numbers would be a real benefit for designers of numerical algorithms and subroutines.

REFERENCES

1. Brent, R. P. A FORTRAN Multiple Precision Package, ACM Transactions on Mathematical Software, Vol. 4, No. 1, March 1978, Pages 57-70.

2. Brent, R. P. MP User's Guide (Third Edition), Technical Report TR-CS-79-08, Department of Computer Science, Australian National University, Box 4, Camberra, ACT 2000, Australia (December 1979).

3. Clenshaw, C. W. and Olver, F. W. J. An Unrestricted Algorithm for the Exponential Function, SIAM J. Numer. Anal., Vol. 17, No. 2, April 1980.

4. Crary, F. D. A Versatile Precompiler for Nonstandard Arithmetics, ACM Transactions on Mathematical Software, Vol. 5, No. 2, June 1979, Pages 204-217.

5. Hull, T. E. Desirable Floating-Point Arithmetic and Elementary Functions for Numerical Computation, Proceedings of the 4th Symposium on Computer Arithmetic, October 25-27, 1978, Santa Monica, Calif., sponsored by the IEEE Computer Society in cooperation with the UCLA Computer Science Department.

6. Knuth, D. E. The Art of Computer Programming, Vol. 2/Seminumerical Algorithms. Addison-Wesley Publishing Co., Reading, Massachusetts, 1969.

7. Lozier, D. W.; Maximon, L. C.; and Sadowski, W. L. Peformance Testing of a FORTRAN Library of Mathematical Function Routines--A Case Study in the Application of Testing Techniques, J. Res. NBS, Vol. 77B, Nos. 3/4, July-Dec. 1973.

8. Maximon, L. C. Fortran Program for Arbitrary Precision Arithmetic, NBS Report 10563, April 1, 1971.

9. Wyatt, W. T., Jr.; Lozier, D. W.; and Orser, D. J. A Portable Extended Precision Arithmetic Package and Library with Fortran Precompiler, ACM Transactions on Mathematical Software, Vol. 2, No. 3, Sept. 1976, Pages 209-231.

APPENDIX A

Summary of useful MP routines, taken from the MP User's Guide (Third Edition), R. P. Brent, The Australian National University.

1.5 Summary of useful MP routines

We mention here the names of those MP routines which are likely to be of interest to someone using the MP package without the aid of the Augment interface. Routines which are called by other MP routines but are unlikely to be called directly by a user of the package are omitted. For the user of Augment, the summary given in Section 5.2 will be more useful. Details of all MP routines may be found in Section 6.

Basic arithmetic

MPADD, MPADDI, MPADDQ, MPDIV, MPDIVI, MPMUL, MPMULI, MPMULQ, MPREC, MPSUB

Powers and roots

MPPWR, MPPWR2, MPQPWR, MPROOT, MPSQRT

Elementary functions

MPASIN, MPATAN, MPATN2, MPCIS, MPCOS, MPCOSH, MPEXP, MPEXP1, MPLG10, MPLN, MPLNI, MPLNS, MPSIN, MPSINH, MPTAN, MPTANH

Special functions

MPBESJ, MPDAW, MPEI, MPERF, MPERFC, MPGAM, MPGAMQ, MPLI, MPLNGM

Constants

MPBERN, MPEPS, MPEUL, MPMAXR, MPMINR, MPPI, MPZETA

Input and output

MPFIN, MPFOUT, MPIN, MPOUT, MPUNFR, MPUNFW

Conversion

MPCAM, MPCDM, MPCIM, MPCMD, MPCMDE, MPCMI, MPCMR, MPCMRE, MPCQM, MPCRM

Comparison

MPCMPA, MPCMPD, MPCMPI, MPCMPQ, MPCMPR, MPCOMP, MPEQ, MPGE, MPGT, MPLE, MPLT, MPNE

General utility routines

MPABS, MPCEIL, MPCHEB, MPCHEV, MPCMF, MPCMIM, MPDIGS, MPDIM, MPFLOR, MPGCDA, MPGCDB, MPINIT, MPMAX, MPMIN, MPMOD, MPNEG, MPPACK, MPPARA, MPPARB, MPPOLY, MPSETR, MPSET2, MPSIGN, MPSTR, MPUNPK

Example and test programs

EXAMPLE, JACOBI, TEST, TEST2

APPENDIX B

The result of processing the runstream shown in Figure 2 on System I.

```
@RUN,N/R LOZ103,35650-LOZIER,CNSLT,2,40,D2000   SEND TO 3RD FLOOR
@ASG,A AMDAI*AUGMENT.
@ASG,A AMDAI*MP.
@XQT AMDAI*AUGMENT.AUGMENT
```

1 1 0 0 A U G M E N T ----- VERSION 4N

```
*DESCRIBE MULTIPAK                                                      MPA00010

COMMENT  AUGMENT DESCRIPTION DECK FOR THE MULTIPLE-PRECISION            MPA00020
         ARITHMETIC PACKAGE OF R. P. BRENT, UNIVAC 1100 VERSION.        MPA00030
         THREE TYPES OF VARIABLE ARE DEFINED HERE -                     MPA00040
               MULTIPLE   (STANDARD MULTIPLE-PRECISION NUMBERS),        MPA00050
               MULTIPAK   (PACKED MULTIPLE-PRECISION NUMBERS), AND      MPA00060
               INITIALIZE (USED ONLY AS A DEVICE TO PERSUADE            MPA00070
                           AUGMENT TO INITIALIZE THE MP PACKAGE).       MPA00080
         WORKING SPACE SHOULD BE ALLOCATED AND THE MP PACKAGE           MPA00090
         INITIALIZED BY THE DECLARATION                                 MPA00100
               INITIALIZE MP                                            MPA00110
         IN THE MAIN PROGRAM.                                           MPA00120
         THIS DESCRIPTION DECK ASSUMES THAT MULTIPLE PRECISION NUMBERS  MPA00130
         WILL HAVE NO MORE THAN 10 DIGITS (BASE 65536) FOR A TOTAL      MPA00140
         PRECISION NOT EXCEEDING ABOUT 43 DECIMAL PLACES.  FOR THIS,    MPA00150
         EACH MP NUMBER REQUIRES 12 WORDS (6 IN PACKED FORMAT).         MPA00160
         SEE COMMENTS IN ROUTINE MPINIT FOR THE METHOD OF CHANGING      MPA00170
         THE PRECISION OR ADAPTING TO A MACHINE WITH WORDLENGTH OTHER   MPA00180
         THAN 36 BITS, AND ALSO REGARDING DECLARATION OF BLANK COMMON.  MPA00190

DECLARE  INTEGER(6), KIND SAFE SUBROUTINE, PREFIX MPK                   MPA00200

SERVICE  COPY(STR)                                                      MPA00210


*DESCRIBE MULTIPLE                                                      MPA00220

DECLARE  INTEGER(16), KIND SAFE SUBROUTINE, PREFIX MP                   MPA00230

OPERATOR + (,NULL UNARY, PRV, @), - (NEG, UNARY),                       MPA00240
         + (ADD, BINARY3, PRV, @, @, @, COMM), * (MUL),                 MPA00250
         - (SUB,,,,,, NONCOMM), / (DIV), ** (PWR2),                     MPA00260
         + (ADDI,,,, INTEGER), * (MULI), / (DIVI), ** (PWR),            MPA00270
         .EQ. (EQ, BINARY2, PRV, @, LOGICAL, COMM), .NE. (NE),          MPA00280
         .GE. (GE,,,,, NONCOMM), .GT. (GT), .LE. (LE), .LT. (LT)        MPA00290

TEST     MPSIGA (SIGA, INTEGER)                                         MPA00300

FIELD    SGN (SIGA, SIGB, (@), INTEGER),                                MPA00310
         EXPON (EXPA, EXPB), BASE (BASA, BASB), NUMDIG (DIGA, DIGB),    MPA00320
         MAXEXP (MEXA, MEXB), DIGIT (DGA, DGB, (@, INTEGER))           MPA00330
```

```
FUNCTION ABS (ABS, ($), $), ASIN (ASIN), ATAN (ATAN), CMF (CMF),          MPA00340
         CMIM (CMIM), COS (COS), COSH (COSH), DAW (DAW), EI (EI),         MPA00350
         ERF (ERF), ERFC (ERFC), EXP (EXP), EXP1 (EXP1), FRAC (CMF),      MPA00360
         GAM (GAM), INT (CMIM), LI (LI), LN (LN), LOG (LN), LNGM (LNGM),  MPA00370
         LNGS (LNGS), LNS (LNS), REC (REC), SIN (SIN), SINH (SINH),       MPA00380
         SQRT (SQRT), TAN (TAN), TANH (TANH),                             MPA00390
         ART1 (ART1, (INTEGER)), LN (LNI), LNI (LNI), LOG (LNI),          MPA00400
         ZETA (ZETA), CAM (CAM), CAM (CAM, (HOLLERITH)),                  MPA00410
         MAX (MAX, ($, $)), MIN (MIN), GCD (GCDA),                        MPA00420
         BESJ (BESJ, ($, INTEGER)), ROOT (ROOT),                          MPA00430
         MPINF (INF(SUBROUTINE),($,INTEGER,INTEGER,HOLLERITH),LOGICAL),   MPA00440
         MPOUTF (OUTF(SUBROUTINE)),                                       MPA00450
         MPINF (INF(SUBROUTINE), ($, INTEGER, INTEGER, INTEGER)),         MPA00460
         MPOUTF (OUTF(SUBROUTINE)),                                       MPA00470
         COMP (COMP, ($, $), INTEGER), CMPA (CMPA),                       MPA00480
         COMP (CMPI, ($, INTEGER)), COMP (CMPR, ($, REAL)),               MPA00490
         ADDQ (ADDQ, ($, INTEGER, INTEGER), $), MULQ (MULQ),              MPA00500
         QPWR (QPWR, (INTEGER, INTEGER, INTEGER, INTEGER)),               MPA00510
         CQM (CQM, (INTEGER, INTEGER)), CTM (CQM),                        MPA00520
         GAM (GAMQ), GAMQ (GAMQ),                                         MPA00530
         BERN (BERN, (INTEGER, INTEGER), MULTIPAK)                        MPA00540

CONVERSION CTM (CDM, DOUBLE PRECISION, $, UPWARD),                        MPA00550
         CTM (CIM, INTEGER), CTM (CRM, REAL),                             MPA00560
         CTM (UNPK, MULTIPAK), CTM (CAM, HOLLERITH),                      MPA00570
         CTD (CMD(SUBROUTINE), $, DOUBLE PRECISION, DOWNWARD),            MPA00580
         CTI (CMI(SUBROUTINE),, INTEGER),                                 MPA00590
         CTR (CMR(SUBROUTINE),, REAL), CTP (PACK,, MULTIPAK)              MPA00600

SERVICE COPY (STR)                                                        MPA00610


*DESCRIBE INITIALIZE                                                     MPA00620

DECLARE INTEGER(1), KIND SAFE SUBROUTINE, PREFIX MPI                      MPA00630

SERVICE COPY (STR), INITIAL (NIT)                                         MPA00640

COMMENT    END OF AUGMENT DESCRIPTION DECK FOR MP PACKAGE                 MPA00650
                                                                          MPA00660


*BEGIN
```

```
:FOR,IS FORM16
      DOUBLE PRECISION RALPHA,RR
      MULTIPLE ALPHA,ANI,AR,B,C,CSB,D,F,FLI,FLK,FLN,G,PI,R,W
      MULTIPLE MPEXP,MPCOS
      DIMENSION F(41),G(41),W(41)
      DIMENSION MPFRA(64)
      INITIALIZE MP
      CALL MPPI(PI)
  111 READ(5,666,END=4000)RALPHA,RR
      ALPHA=RALPHA
      R=RR
      WRITE(6,777)RALPHA,RR
  666 FORMAT(2F8.4)
  777 FORMAT(1X,6HALPHA=,F8.4,10X,2HR=,F8.4)
      AR=ALPHA*R
      DO 77 N=20,40,10
      NM1=N-1
      NP1=N+1
      FLN=N
      D=PI*ALPHA**3*MPEXP(-AR)/(2.D0*AR*FLN)
C ===== MIXED MODE OPERANDS ACCEPTED =====
      F(1)=1.D0
      F(2)=1.D0-2.D0*AR
C ===== MIXED MODE OPERANDS ACCEPTED =====
C ===== MIXED MODE OPERANDS ACCEPTED =====
      DO 10 K=1,NM1
      FLK=K
      F(K+2)=((2.D0*FLK+1.D0-2.D0*AR)*F(K+1)-FLK*F(K))/(FLK+1.D0)
C ===== MIXED MODE OPERANDS ACCEPTED =====
C ===== MIXED MODE OPERANDS ACCEPTED =====
C ===== MIXED MODE OPERANDS ACCEPTED =====
C ===== MIXED MODE OPERANDS ACCEPTED =====
   10 CONTINUE
      DO 3 KK=1,NP1
      CALL MPOUTE(F(KK),MPFRA,MPCHR,64)
      WRITE(6,1000)KK,MPFRA,MPCHR
    3 CONTINUE
      WRITE(6,2000)
      G(1)=0.D0
      G(2)=-6.D0*AR+2.D0*AR*AR
C ===== MIXED MODE OPERANDS ACCEPTED =====
C ===== MIXED MODE OPERANDS ACCEPTED =====
      DO 20 K=2,NM1
      G(K+1)=F(K+2)+F(K+1)-F(K)-F(K-1)
   20 CONTINUE
      DO 5 KK=1,N
      CALL MPOUTE(G(KK),MPFRA,MPCHR,64)
      WRITE(6,1000)KK,MPFRA,MPCHR
    5 CONTINUE
      WRITE(6,2000)
      W(1)=0.D0
      DO 100 I=0,N
      FLI=I
      B=FLI*PI/FLN
      CSB=MPCOS(B)
      W(2)=2.D0*(1.D0+CSB-AR)
C ===== MIXED MODE OPERANDS ACCEPTED =====
C ===== MIXED MODE OPERANDS ACCEPTED =====
```

```
                                        DO 30 K=1,NM1
                                        W(K+2)=2.D0*CSB*W(K+1)-W(K)+G(K+1)
C ===== MIXED MODE OPERANDS ACCEPTED =====
                                     30 CONTINUE
                                        ANI=1.D0
                                        IF(I.EQ.0)ANI=0.5D0
                                        IF(I.EQ.N)ANI=0.5D0
                                        C=ANI*(-1)**I*D
                                        W(N+1)=C*W(N+1)
                                        CALL MPOUTE(W(N+1),MPFRA,MPCHR,64)
                                        WRITE(6,1000)I,MPFRA,MPCHR
                                    100 CONTINUE
                                        WRITE(6,2000)
                                     77 CONTINUE
                                        GO TO 111
                                   1000 FORMAT(1X,I3,5X,64A1,'E',I3)
                                   2000 FORMAT(1H0)
                                   3000 FORMAT(1X,2D30.18)
                                   4000 CONTINUE
                                        STOP
                                        END
```

==========

```
              *END
@FOR,IS FORM16
FOR S 4R1   E -05/16/80-18:49:26 (,0)

   MAIN PROGRAM

   STORAGE USED: CODE(1) 000743; DATA(0) 004402; BLANK COMMON(2) 000000


   EXTERNAL REFERENCES (BLOCK, NAME)

    0003   MPINIT
    0004   MPPI
    0005   MPCDM
    0006   MPMUL
    0007   MPCIM
    0010   MPPWR
    0011   MPNEG
    0012   MPEXP
    0013   MPDIV
    0014   MPSUB
    0015   MPADD
    0016   MPOUTE
    0017   MPCOS
    0020   MPMULI
    0021   NINTFS
    0022   NRDUS
    0023   NIO2S
    0024   NWDUS
    0025   NIO3S
    0026   NIO1S
    0027   XPII
    0030   NSTOPS
```

```
0027   XPII
0030   NSTOPS


STORAGE ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

0000   004344 1000F     0001   000010 111L     0001   000047 127G     0001   000177 153G     0001   000314 176G
0000   004350 2000F     0001   000415 223G     0001   000452 233G     0001   000513 251G     0001   000567 265G
0000   004351 3000F     0001   000737 4000L    0000   004334 666F     0000   004336 777F     0000 I 000153 ALPHA
0000 I 000173 ANI       0000 I 000213 AR       0000 I 000233 B        0000 I 000253 C        0000 I 000273 CSB
0000 I 000313 D         0000 I 000333 F        0000 I 001553 FLI      0000 I 001573 FLK      0000 I 001613 FLN
0000 I 001633 G         0000 I 000040 I        0000 I 000041 K        0000 I 000042 KK       0000 I 004333 MP
0000 I 000043 MPCHR     0000 I 000044 MPFRA    0000 I 000000 MPTMP    0000 I 000144 N        0000 I 000145 NM1
0000 I 000146 NP1       0000 I 003053 PI       0000 I 003073 R        0000 D 000147 RALPHA   0000 D 000151 RR
0000 I 003113 W


00100      1*      C                ===== PROCESSED BY AUGMENT, VERSION 4N =====                    000000
00100      2*      C                ----- TEMPORARY STORAGE LOCATIONS -----                          000000
00100      3*      C                      MULTIPLE                                                  000000
00101      4*            INTEGER MPTMP(16,2)                                                       000000
00101      5*      C                ----- LOCAL VARIABLES -----                                     000000
00103      6*            INTEGER I, K, KK, MPCHR, MPFRA(64), N, NM1, NP1                           000001
00104      7*            DOUBLE PRECISION RALPHA, RR                                               000001
00104      8*      C                      MULTIPLE                                                  000001
00105      9*            INTEGER ALPHA(16), ANI(16), AR(16), B(16), C(16), CSB(16), D(16),         000001
00105     10*           *     F(16,41), FLI(16), FLK(16), FLN(16), G(16,41), PI(16), R(16),       000001
00105     11*           *     W(16,41)                                                            000001
00105     12*      C                      INITIALIZE                                                000001
00106     13*            INTEGER MP(1)                                                             000001
00106     14*      C                ===== TRANSLATED PROGRAM =====                                  000001
00106     15*      C                ----- BEGIN INITIALIZATION -----                                000001
00107     16*            CALL MPINIT (MP)                                                          000001
00107     17*      C                ----- END INITIALIZATION -----                                  000001
00110     18*            CALL MPPI(PI)                                                             000004
00111     19*          111 READ(5,666,END=4000)RALPHA,RR                                           000010
00115     20*            CALL MPCDM (RALPHA,ALPHA)                                                 000017
00116     21*            CALL MPCDM (RR,R)                                                         000023
00117     22*            WRITE(6,777)RALPHA,RR                                                     000027
00123     23*          666 FORMAT(2F8.4)                                                           000036
00124     24*          777 FORMAT(1X,6HALPHA=,F8.4,10X,2HR=,F8.4)                                  000036
00125     25*            CALL MPMUL (ALPHA,R,AR)                                                   000036
00126     26*            DO 77    N=20,40,10                                                       000047
00131     27*            NM1=N-1                                                                   000052
00132     28*            NP1=N+1                                                                   000057
00133     29*            CALL MPCIM (N,FLN)                                                        000062
00133     30*      C ===== MIXED MODE OPERANDS ACCEPTED =====                                      000062
00134     31*            CALL MPPWR (ALPHA,3,MPTMP(1,1))                                           000072
00135     32*            CALL MPMUL (PI,MPTMP(1,1),MPTMP(1,1))                                     000100
00136     33*            CALL MPNEG (AR,MPTMP(1,2))                                                000105
00137     34*            CALL MPEXP (MPTMP(1,2),MPTMP(1,2))                                        000111
00140     35*            CALL MPMUL (MPTMP(1,1),MPTMP(1,2),MPTMP(1,2))                             000115
00141     36*            CALL MPCDM (2.D0,MPTMP(1,1))                                              000122
00142     37*            CALL MPMUL (MPTMP(1,1),AR,MPTMP(1,1))                                     000126
00143     38*            CALL MPMUL (MPTMP(1,1),FLN,MPTMP(1,1))                                    000133
00144     39*            CALL MPDIV (MPTMP(1,2),MPTMP(1,1),D)                                      000140
```

```
00145    40*          CALL MPCDM (1.D0,F(1,1))                                    000145
00145    41*     C ===== MIXED MODE OPERANDS ACCEPTED =====                       000145
00145    42*     C ===== MIXED MODE OPERANDS ACCEPTED =====                       000145
00146    43*          CALL MPCDM (2.D0,MPTMP(1,1))                                000151
00147    44*          CALL MPMUL (MPTMP(1,1),AR,MPTMP(1,1))                       000155
00150    45*          CALL MPCDM (1.D0,MPTMP(1,2))                                000162
00151    46*          CALL MPSUB (MPTMP(1,2),MPTMP(1,1),F(1,2))                   000166
00152    47*          DO 10     K=1,NM1                                           000177
00155    48*          CALL MPCIM (K,FLK)                                          000177
00155    49*     C ===== MIXED MODE OPERANDS ACCEPTED =====                       000177
00155    50*     C ===== MIXED MODE OPERANDS ACCEPTED =====                       000177
00155    51*     C ===== MIXED MODE OPERANDS ACCEPTED =====                       000177
00155    52*     C ===== MIXED MODE OPERANDS ACCEPTED =====                       000177
00156    53*          CALL MPCDM (2.D0,MPTMP(1,1))                                000203
00157    54*          CALL MPMUL (MPTMP(1,1),FLK,MPTMP(1,1))                      000207
00160    55*          CALL MPCDM (1.D0,MPTMP(1,2))                                000214
00161    56*          CALL MPADD (MPTMP(1,1),MPTMP(1,2),MPTMP(1,2))               000220
00162    57*          CALL MPCDM (2.D0,MPTMP(1,1))                                000225
00163    58*          CALL MPMUL (MPTMP(1,1),AR,MPTMP(1,1))                       000231
00164    59*          CALL MPSUB (MPTMP(1,2),MPTMP(1,1),MPTMP(1,1))               000236
00165    60*          CALL MPMUL (MPTMP(1,1),F(1,K+1),MPTMP(1,1))                 000243
00166    61*          CALL MPMUL (FLK,F(1,K),MPTMP(1,2))                          000252
00167    62*          CALL MPSUB (MPTMP(1,1),MPTMP(1,2),MPTMP(1,2))               000261
00170    63*          CALL MPCDM (1.D0,MPTMP(1,1))                                000266
00171    64*          CALL MPADD (FLK,MPTMP(1,1),MPTMP(1,1))                      000272
00172    65*          CALL MPDIV (MPTMP(1,2),MPTMP(1,1),F(1,K+2))                 000277
00173    66*       10 CONTINUE                                                    000314
00175    67*          DO 3      KK=1,NP1                                          000314
00200    68*          CALL MPOUTE(F(1,KK),MPFRA,MPCHR,64)                         000314
00201    69*          WRITE(6,1000)KK,MPFRA,MPCHR                                 000324
00205    70*        3 CONTINUE                                                    000341
00210    71*          WRITE(6,2000)                                               000341
00212    72*          CALL MPCDM (0.D0,C(1,1))                                    000346
00212    73*     C ===== MIXED MODE OPERANDS ACCEPTED =====                       000346
00212    74*     C ===== MIXED MODE OPERANDS ACCEPTED =====                       000346
00213    75*          CALL MPCDM (6.D0,MPTMP(1,1))                                000352
00214    76*          CALL MPMUL (MPTMP(1,1),AR,MPTMP(1,1))                       000356
00215    77*          CALL MPNEG (MPTMP(1,1),MPTMP(1,1))                          000363
00216    78*          CALL MPCDM (2.D0,MPTMP(1,2))                                000367
00217    79*          CALL MPMUL (MPTMP(1,2),AR,MPTMP(1,2))                       000373
00220    80*          CALL MPMUL (MPTMP(1,2),AR,MPTMP(1,2))                       000400
00221    81*          CALL MPADD (MPTMP(1,1),MPTMP(1,2),G(1,2))                   000405
00222    82*          DO 20     K=2,NM1                                           000415
00225    83*          CALL MPADD (F(1,K+2),F(1,K+1),MPTMP(1,1))                   000415
00226    84*          CALL MPSUB (MPTMP(1,1),F(1,K),MPTMP(1,1))                   000425
00227    85*          CALL MPSUB (MPTMP(1,1),F(1,K-1),G(1,K+1))                   000434
00230    86*       20 CONTINUE                                                    000452
00232    87*          DO 5      KK=1,N                                            000452
00235    88*          CALL MPOUTE(G(1,KK),MPFRA,MPCHR,64)                         000452
00236    89*          WRITE(6,1000)KK,MPFRA,MPCHR                                 000462
00243    90*        5 CONTINUE                                                    000477
00245    91*          WRITE(6,2000)                                               000477
00247    92*          CALL MPCDM (0.D0,W(1,1))                                    000504
00250    93*          DO 100    I=0,N                                             000513
00253    94*          CALL MPCIM (I,FLI)                                          000513
00254    95*          CALL MPMUL (FLI,PI,MPTMP(1,1))                              000517
00255    96*          CALL MPDIV (MPTMP(1,1),FLN,B)                               000524
00256    97*          CALL MPCOS (B,CSB)                                          000531
00256    98*     C ===== MIXED MODE OPERANDS ACCEPTED =====                       000531
00256    99*     C ===== MIXED MODE OPERANDS ACCEPTED =====                       000531
```

```
00256     98*      C ===== MIXED MODE OPERANDS ACCEPTED =====                    000531
00256     99*      C ===== MIXED MODE OPERANDS ACCEPTED =====                    000531
00257    100*            CALL MPCDM (1.D0,MPTMP(1,1))                           000535
00260    101*            CALL MPADD (MPTMP(1,1),CSB,MPTMP(1,1))                 000541
00261    102*            CALL MPSUB (MPTMP(1,1),AR,MPTMP(1,1))                  000546
00262    103*            CALL MPCDM (2.D0,MPTMP(1,2))                           000553
00263    104*            CALL MPMUL (MPTMP(1,2),MPTMP(1,1),W(1,2))              000557
00264    105*            DO 30     K=1,NM1                                      000567
00264    106*      C ===== MIXED MODE OPERANDS ACCEPTED =====                    000567
00267    107*            CALL MPCDM (2.D0,MPTMP(1,1))                           000567
00270    108*            CALL MPMUL (MPTMP(1,1),CSB,MPTMP(1,1))                 000572
00271    109*            CALL MPMUL (MPTMP(1,1),W(1,K+1),MPTMP(1,1))            000577
00272    110*            CALL MPSUB (MPTMP(1,1),W(1,K),MPTMP(1,1))              000606
00273    111*            CALL MPADD (MPTMP(1,1),C(1,K+1),W(1,K+2))              000615
00274    112*         30 CONTINUE                                               000627
00276    113*            CALL MPCDM (1.D0,ANI)                                  000627
00277    114*            IF (I.EQ.0) CALL MPCDM (0.5D0,ANI)                     000633
00301    115*            IF (I.EQ.N) CALL MPCDM (0.5D0,ANI)                     000641
00303    116*            CALL MPMULI (ANI,(-1)**I,MPTMP(1,1))                   000650
00304    117*            CALL MPMUL (MPTMP(1,1),D,C)                            000662
00305    118*            CALL MPMUL (C,W(1,N+1),W(1,N+1))                       000667
00306    119*            CALL MPOUTE(W(1,N+1),MPFRA,MPCHR,64)                   000700
00307    120*            WRITE(6,1000)I,MPFRA,MPCHR                             000710
00314    121*          100 CONTINUE                                             000725
00316    122*            WRITE(6,2000)                                          000725
00320    123*           77 CONTINUE                                             000735
00322    124*            GO TO 111                                              000735
00323    125*         1000 FORMAT(1X,I3,5X,64A1,'E',I3)                         000737
00324    126*         2000 FORMAT(1H0)                                          000737
00325    127*         3000 FORMAT(1X,2D30.18)                                   000737
00326    128*         4000 CONTINUE                                             000737
00327    129*            STOP                                                   000737
00330    130*            END                                                    000742

       END OF COMPILATION:        NO  DIAGNOSTICS.

@MAP,IS ,FORMI6
MAP 29R1 SL73R1 05/16/80 18:49:34
    1.              IN FORMI6
    2.              IN AMDAI*MP.MPINIT/MP16
    3.              LIB AMDAI*MP.
    4.              END

AFCM STATUS OF OUTPUT ELEMENT=UNKNOWN


ADDRESS LIMITS    001000 030042      11811 IBANK WORDS DECIMAL
                  040000 054132       6235 DBANK WORDS DECIMAL
STARTING ADDRESS  027061



            SEGMENT SMAIN3          001000 030042     040000 054132

NSWTC3/FOR4R1-E          S(1)     001000 001030                  27 JUL 78   17:07:41
NRBLK3/FOR-E2            S(1)     001031 001053                  29 APR 74   13:48:27
```

NEXP3$/FOR68	$(1)	001054 001141	$(2)	040000 040007	10 JUL 72	21:40:28
NINTR$/NBS	$(1)	001142 001421	$(2)	040010 040143	03 MAY 79	13:03:09
EXP$/FOR59	$(1)	001422 001511	$(2)	040144 040164	12 MAY 71	15:49:18
NWEF$/FOR4R1-E	$(1)	001512 001717	$(2)	040165 040204	03 JAN 79	14:33:58
NRWND$/FOR-E3	$(1)	001720 002003	$(2)	040205 040216	23 JUN 75	10:07:02
NCLOS$/FOR4R1-E	$(1)	002004 002247	$(2)	040217 040244	27 JUL 78	17:00:43
SQRT$/FOR59	$(1)	002250 002310	$(2)	040245 040256	12 MAY 71	15:54:25
NFTCH$/FOR4R1-E	$(1)	002311 002576	$(2)	040257 040272	27 JUL 78	17:03:29
NINPT$/FOR4R1-E	$(1)	002577 004223	$(2)	040273 040326	27 JUL 78	17:04:45
NININ$/FOR4R1-E	$(1)	004224 004471	$(2)	040327 040327	27 JUL 78	17:03:59
NIBUF$/FOR-E2	$(1)	004472 004531	$(2)	040330 040330	29 APR 74	13:47:56
NOTIN$/FOR4R1-E	$(1)	004532 005023	$(2)	040331 040334	31 AUG 78	16:01:59
NBSBL$/FOR-E3	$(1)	005024 005060			16 APR 75	13:13:54
NBFOO$			$(2)	040335 042562	23 JUN 75	09:58:12
NUPDA$/FOR68	$(1)	005061 005114			10 JUL 72	21:41:26
NTAB$/FOR			$(2)	042563 042651	21 APR 76	11:50:31
NWBLK$/FOR68	$(1)	005115 005226			10 JUL 72	21:41:28
NFCHK$/FOR4R1-E	$(1)	005227 006315	$(2)	042652 043024	10 AUG 78	13:00:55
	$(3)	006316 006316	$(4)	043025 043076		
FORCOM$/FORFTN			$(2)	043077 043104	31 AUG 78	14:44:53
NCNVT$/FOR4R1-E	$(1)	006317 006540	$(2)	043105 043201	27 JUL 78	17:00:55
NFTV$/FOR-E2	$(1)	006541 006563			29 APR 74	13:47:54
NFNT$/FOR-E3	$(1)	006564 007446	$(2)	043202 043256	16 APR 75	13:20:52
NBDCV$/FOR4R1-E	$(1)	007447 007601	$(2)	043257 043335	27 JUL 78	17:00:26
NOUT$/FOR4R1-E	$(1)	007602 011332	$(2)	043336 043400	27 JUL 78	17:06:34
NIGER$/FOR4R1-E	$(1)	011333 011554	$(2)	043401 043547	27 JUL 78	17:05:06
NOBUF$/FOR63	$(1)	011555 011615			10 JUL 72	21:41:08
SINCOS$/FOR-E3	$(1)	011616 011752	$(2)	043550 043572	17 APR 75	09:52:54
NEXP1$/FOR68	$(1)	011753 012010	$(0)	043573 043573	10 JUL 72	21:40:25
NERCOM$/FOR-TE3	$(1)	012011 012070	$(2)	043574 043607	11 MAR 75	15:33:59
ERU$/SYS74R1					20 DEC 78	17:30:56
NERR$/FOR4R1-E	$(1)	012071 012436	$(2)	043610 043770	27 JUL 78	17:01:50
ALOC$/FOR-E3	$(1)	012437 012555	$(2)	043771 044031	17 APR 75	09:42:35
FORVCOM$/FOR4R1			$(2)	044032 044041	28 JUL 78	14:36:41
NSTOP$/FOR4R1	$(1)	012556 012621	$(2)	044042 044051	28 JUL 78	14:38:58
NIER$/FOR4R1-E	$(1)	012622 013003	$(2)	044052 044172	27 JUL 78	17:03:43
MPADDQ	$(1)	013004 013052	$(0)	044173 044201	08 NOV 78	19:44:30
			$(2)	BLANK$COMMON		
MPSIN	$(1)	013053 013503	$(0)	044202 044247	08 NOV 78	20:41:30
			$(2)	BLANK$COMMON		
MPCRM	$(1)	013504 014006	$(0)	044250 044303	08 NOV 78	20:00:03
			$(2)	BLANK$COMMON		
MPNLP	$(1)	014007 014050	$(0)	044304 044320	08 NOV 78	20:36:54
			$(2)	BLANK$COMMON		
MPCMI	$(1)	014051 014246	$(0)	044321 044345	08 NOV 78	19:53:33
			$(2)	BLANK$COMMON		
MPCMP	$(1)	014247 014371	$(0)	044346 044367	08 NOV 78	19:59:06
			$(2)	BLANK$COMMON		
MPADDI	$(1)	014372 014435	$(0)	044370 044376	08 NOV 78	19:44:27
			$(2)	BLANK$COMMON		
MPCLR	$(1)	014436 014501	$(0)	044377 044411	08 NOV 78	19:53:07
			$(2)	BLANK$COMMON		
MPCHF	$(1)	014502 014631	$(0)	044412 044435	08 NOV 78	19:53:30
			$(2)	BLANK$COMMON		
MPABS	$(1)	014632 014660	$(0)	044436 044443	08 NOV 78	19:41:40
			$(2)	BLANK$COMMON		
MPSIN1	$(1)	014661 015211	$(0)	044444 044477	08 NOV 78	20:41:54
			$(2)	BLANK$COMMON		
MPCHK	$(1)	015212 015400	$(0)	044500 044661	08 NOV 78	19:50:55
			$(2)	BLANK$COMMON		

MPCHK	$(1)	015212 015400	$(0)	044500 044661	08 NOV 78	19:50:55
			$(2)	BLANK$COMMON		
MPNEG	$(1)	015401 015427	$(0)	044662 044667	08 NOV 78	20:37:35
			$(2)	BLANK$COMMON		
MPEXP1	$(1)	015430 016016	$(0)	044670 044733	08 NOV 78	20:16:39
			$(2)	BLANK$COMMON		
MPCOS	$(1)	016017 016153	$(0)	044734 044747	08 NOV 78	19:59:40
			$(2)	BLANK$COMMON		
MPMULQ	$(1)	016154 016270	$(0)	044750 044775	08 NOV 78	20:37:17
			$(2)	BLANK$COMMON		
MPMULI	$(1)	016271 016315	$(0)	044776 045002	08 NOV 78	20:37:15
			$(2)	BLANK$COMMON		
MPNZR	$(1)	016316 016633	$(0)	045003 045062	08 NOV 78	20:37:38
			$(2)	BLANK$COMMON		
MPREC	$(1)	016634 017161	$(0)	045063 045160	08 NOV 78	20:38:51
			$(2)	BLANK$COMMON		
MPMUL	$(1)	017162 017475	$(0)	045161 045250	08 NOV 78	20:37:01
			$(2)	BLANK$COMMON		
MPSUB	$(1)	017476 017530	$(0)	045251 045257	08 NOV 78	20:42:02
			$(2)	BLANK$COMMON		
MPCMPR	$(1)	017531 017573	$(0)	045260 045267	08 NOV 78	19:57:51
			$(2)	BLANK$COMMON		
MPERFL(COMMONBLOCK)				045270 045270		
MPERR	$(1)	017574 017616	$(0)	045271 045311	25 FEB 80	11:21:44
	$(3)	MPERFL	$(2)	BLANK$COMMON		
MPCMP1	$(1)	017617 017661	$(0)	045312 045321	08 NOV 78	19:53:39
			$(2)	BLANK$COMMON		
MPOUT2	$(1)	017662 020763	$(0)	045322 045441	08 NOV 78	20:37:53
			$(2)	BLANK$COMMON		
MPSTR	$(1)	020764 021061	$(0)	045442 045462	08 NOV 78	20:42:00
			$(2)	BLANK$COMMON		
MPEXP	$(1)	021062 021641	$(0)	045463 045544	08 NOV 78	20:14:14
			$(2)	BLANK$COMMON		
MPCMEF	$(1)	021642 022223	$(0)	045545 045612	08 NOV 78	19:53:28
			$(2)	BLANK$COMMON		
MPSET	$(1)	022224 022453	$(0)	045613 045711	08 NOV 78	20:38:59
			$(2)	BLANK$COMMON		
MPCHR	$(1)	022454 022635	$(0)	045712 045747	08 NOV 78	19:57:53
			$(2)	BLANK$COMMON		
MPDIV1	$(1)	022636 023332	$(0)	045750 046035	08 NOV 78	20:01:15
			$(2)	BLANK$COMMON		
MPEXT	$(1)	023333 023452	$(0)	046036 046050	08 NOV 78	20:17:14
			$(2)	BLANK$COMMON		
MPMAXR	$(1)	023453 023522	$(0)	046051 046066	08 NOV 78	20:36:43
			$(2)	BLANK$COMMON		
MPOVFL	$(1)	023523 023553	$(0)	046067 046105	08 NOV 78	20:37:55
			$(2)	BLANK$COMMON		
MPCCD	$(1)	023554 023645	$(0)	046106 046115	08 NOV 78	20:19:16
			$(2)	BLANK$COMMON		
MPCOM	$(1)	023646 023734	$(0)	046116 046132	08 NOV 78	19:59:51
			$(2)	BLANK$COMMON		
MPART1	$(1)	023735 024216	$(0)	046133 046163	08 NOV 78	19:49:20
			$(2)	BLANK$COMMON		
MPP1	$(1)	024217 024334	$(0)	046164 046212	08 NOV 78	20:38:06
			$(2)	BLANK$COMMON		
MPUNFL	$(1)	024335 024357	$(0)	046213 046222	08 NOV 78	20:42:11
			$(2)	BLANK$COMMON		
MPDIV	$(1)	024360 024543	$(0)	046223 046262	08 NOV 78	20:01:11

```
                                                    $(2)    BLANK$COMMON
MPOUT                       $(1)    024544 024573   $(0)    046263 046267          08 NOV 78   20:37:40
                                                    $(2)    BLANK$COMMON
MPOUTE                      $(1)    024574 024751   $(0)    046270 046315          08 NOV 78   20:37:42
                                                    $(2)    BLANK$COMMON
MPMUL2                      $(1)    024752 025355   $(0)    046316 046403          08 NOV 78   20:37:22
                                                    $(2)    BLANK$COMMON
MPCIM                       $(1)    025356 025456   $(0)    046404 046423          08 NOV 78   19:51:55
                                                    $(2)    BLANK$COMMON
MPADD3                      $(1)    025457 026053   $(0)    046424 046450          08 NOV 78   19:49:14
                                                    $(2)    BLANK$COMMON
MPADD2                      $(1)    026054 026322   $(0)    046451 046522          08 NOV 78   19:48:57
                                                    $(2)    BLANK$COMMON
MPPWR                       $(1)    026323 026526   $(0)    046523 046562          08 NOV 78   20:38:38
                                                    $(2)    BLANK$COMMON
MPADD                       $(1)    026527 026555   $(0)    046563 046567          08 NOV 78   19:41:42
                                                    $(2)    BLANK$COMMON
MPCDM                       $(1)    026556 027060   $(0)    046570 046632          07 MAR 80   11:36:16
                                                    $(2)    BLANK$COMMON
                                                            046633 047520
BLANK$COMMON(COMMONBLOCK)
FORM16                      $(1)    027061 030023   $(0)    047521 054122          16 MAY 80   18:49:33
                                                    $(2)    BLANK$COMMON
MPINIT/MP16                 $(1)    030024 030042   $(0)    054123 054132          16 MAR 80   09:34:57
                                                    $(2)    BLANK$COMMON

SYS$*RLIBS. LEVEL
END MAP. ERRORS : NONE

$XQT FORM16
 ALPHA=    .5000          R=  1.0000
   1      1.0000000000000000000000000000000000000000000000000000000000000000E  0
   2      0.0000000000000000000000000000000000000000000000000000000000000000E  0
   3     -5.0000000000000000000000000000000000000000000000000000000000000000E -1
   4     -6.6666666666666666666666666666666666666666666666666666666666666667E -1
   5     -6.2500000000000000000000000000000000000000000000000000000000000000E -1
   6     -4.6666666666666666666666666666666666666666666666666666666666666667E -1
   7     -2.5694444444444444444444444444444444444444444444444444444444444444E -1
   8     -4.0476190476190476190476190476190476190476190476190476190476190476E -2
   9      1.5399305555555555555555555555555555555555555555555555555555555556E -1
  10      3.0974426807760141093474426807760141093474426807760141093474427E -1
  11      4.1894593253968253968253968253968253968253968253968253968253968E -1
  12      4.8013417909251242584575917909251242584575917909251242584575918E -1
  13      4.9621222350823045267489711934156378600822045267489711934156379E -1
  14      4.7288332423749090415757082423749090415757082423749090415757082E -1
  15      4.1744339461198340166594134848103102071356039610007863976117944E -1
  16      3.3786990065404417256269108120959972811824663676515528367380219E -1
  17      2.4215288127759838449322576306703290830274957259084243211227338E -1
  18      1.3782198767167303192824512463479161269388471380379254640070077E -1
  19      3.1630477728761697176416459191297521691407640849145846206398822E -2
  20     -7.0636767360773408609168269757651709262763040999481774622240017E -2
  21     -1.6425881182779295995333760748568647036676223659659009107430451E -1


   1      0.6000000000000000000000000000000000000000000000000000000000000000E  0
   2     -2.5000000000000000000000000000000000000000000000000000000000000000E  0
   3     -2.1666666666666666666666666666666666666666666666666666666666666667E  0
   4     -7.9166666666666666666666666666666666666666666666666666666666666667E -1
   5      7.5000000000000000000000000000000000000000000000000000000000000001E -2
   6      5.6805555555555555555555555555555555555555555555555555555555555556E -1
   7      7.9424603174603174603174603174603174603174603174603174603174603E -1
   8      8.3712797619047619047619047619047619047619047619047619047619047E -1
```

7	7.9424603174603174603174603174603174603174603174603174603E -1
8	8.3712797619047619047619047619047619047619047619047619048E -1
9	7.6115795855379188712522045855379188712522045855379188712522046E -1
10	6.1517333553791887125220458553791887125220458553791887125220459E -1
11	4.3534278799037999037999037999037999037999037999037999037999904E -1
12	2.4765620198345892790337234781679226123670568115012559457003901E -1
13	7.0015436113526391304169081946859724637502415280193057970835748E -2
14	-8.6019683751268572697144125715554286982858411429840001268572698E -2
15	-2.1378225247969378260383551388842394133399424404715410006415297E -1
16	-3.1030393691783174876759532844188928845013501098157182813267469E -1
17	-3.7533842631675615780716154198880622783517274647059894492200748E -1
18	-4.1057031653120782795125526045054350203570385470305932317897598E -1
19	-4.1898115858128306010597125548629217023150294964543730976879935E -1
20	-4.0434804458900102991891601828754077567833089534947666114362813E -1

0	4.4041070371375241719317506779367576090993504634237209731260529E -2
1	-1.0836659059255916781191071031301976356548989622742134209760114E -1
2	-5.9690490598945944258903582720305775163761266027855447898479509E -2
3	1.6610281250832360136298922803268128160707160813500398412919040E -1
4	1.0810311906267657259887274263725397813335999860338080534722257E -1
5	9.6637284677011177565415365713963880868878802466983070936410467E -2
6	5.8455219411583004833555498371804489730023579762816061402571625E -2
7	5.0863648066458313205493138372011184427779665041234330089705261E -2
8	3.1043605421103477786275762446256026627476860228302409217168554E -2
9	2.8084334079409584719805826417080404905801880347583688759528993E -2
10	1.6575544114300579783964312303986947479582232015234121094657025E -2
11	1.5833286563630234224740206668419590080422632956824453359975942E -2
12	8.4259174130623032578815301801318272092577821845257011310351711E -3
13	8.7978124231713606799903068664548619192757876566919862346357517E -3
14	3.6346345052507107350069947229244576588822938260937879534217361E -3
15	4.6532240707688418971008474325507559431752686045649657607559035E -3
16	8.2987750779958258263907165026643621366061137026157914714510561E -4
17	2.3088856652403367682208921632041320484768459107009558702760439E -3
18	-6.5117041671945424231616313962519877691983332363637293792489257E -4
19	1.2435233830397564572088426421332564022691275462275118233638492E -3
20	-5.5748190343512964201667730100465286191131018524350893940606678E -4

1	1.0000000000000000000000000000000000000000000000000000000000006E 0
2	0.0000000000000000000000000000000000000000000000000000000000000E 0
3	-5.0000000000000000000000000000000000000000000000000000000000000E -1
4	-6.6666666666666666666666666666666666666666666666666666666666667E -1
5	-6.2500000000000000000000000000000000000000000000000000000000000E -1
6	-4.6666666666666666666666666666666666666666666666666666666666667E -1
7	-2.5694444444444444444444444444444444444444444444444444444444444E -1
8	-4.0476190476190476190476190476190476190476190476190476190476196E -2
9	1.5399305555555555555555555555555555555555555555555555555555556E -1
10	3.0974426807760141093474426807760141093474426807760141093474427E -1
11	4.1894593253968253968253968253968253968253968253968253968253968E -1
12	4.8913417909251242584575917909251242584575917909251242584575918E -1
13	4.9621222350823045267489711934156378600823045267489711934156379E -1
14	4.7288338242374909041575708242374909041575708242374909041575708 2E -1
15	4.1744339461190340166594134848103102071356039610007863976117944E -1
16	3.3786990065404417256269108120959972811324663676515528367330219E -1
17	2.4215288127759688449322576306703290830274957259084243211227038E -1
18	1.3782198767167303192824512463479161269388471380379254640070077E -1
19	3.1630477728761697176416459191297521691407640849145846206398822E -2

```
20     -7.0636767360773340860916826975765170926276304099948177462224017E -2
21     -1.6425881182779295995333760748568647036676223659659009107430451E -1
22     -2.4560031551386912290072237428153120934023635151736381398703334E -1
23     -3.1208178197312959103682935284658613248126908160004037613159116E -1
24     -3.6210349849576440442628702135626535755176695030520750617718599E -1
25     -3.9495333105929925046675532777669689167967045155160902638016494E -1
26     -4.1069103707792073253173468883500328877527099468609012471901814E -1
27     -4.1002763759282905255453274028202854410660724790439540678795321E -1
28     -3.9420259965930265507446668833168143649579744552556362630586279E -1
29     -3.6486407737842710553745775651057238856766665589149142275863764E -1
30     -3.2395570837004977820732989970017150130713837776578407372136379E -1
31     -2.736124280496167025146286414601115935783891829854075052795360E -1


 1      0.000000000000000000000000000000000000000000000000000000000000E  0
 2     -2.500000000000000000000000000000000000000000000000000000000000E  0
 3     -2.166666666666666666666666666666666666666666666666666666666667E  0
 4     -7.916666666666666666666666666666666666666666666666666666666667E -1
 5      7.500000000000000000000000000000000000000000000000000000000001E -2
 6      5.680555555555555555555555555555555555555555555555555555555556E -1
 7      7.942460317460317460317
```

12	2.40682567454372125382259553014488577635117093771574748572643 42E -2
13	1.76907116177237261753053532816780459764156179794382248394797 27E -2
14	1.62340191265617062825723637965547426637475068146090882471136 76E -2
15	1.17429073391020255145276344367659391427623113459334935817939 44E -2
16	1.09878824272282582015197608531101643167738909353156038264561 87E -2
17	7.68444526358614225186661228499174129080620674328605947415082 49E -3
18	7.37305883149734133273742801493038210209768458547916527777447 65E -3
19	4.85298440432094713003440654110631930327124652901116844112294 38E -3
20	4.83789896010544123705764699512158090484288371149975091557639 40E -3
21	2.85679258509713683247161493620449435710847951203317207995411 79E -3
22	3.05342203570966948981478899983063220625879957991716622076735 85E -3
23	1.45765512736215529877752076753999750394239341568030927678084 23E -3
24	1.81829254834097821852062903766902119966081571545007908683472 72E -3
25	5.08268232095054451024931803561060271130236445707888162912301 36E -4
26	1.00826457110874473234845495491048588336366571255728060400093 89E -3
27	-8.18135557400861654156148114476317985825163603904913744219030 10E -5
28	5.48701277123420263527998208530985791718043021714775963752572 49E -4
29	-3.64860499398975593934172269011544646865371491784212765540682 72E -4
30	1.99849425762753982183546691271651199745385001100014511568323 34E -4

1	1.00000000000000000000000000000000000000000000000000000000000000E 0
2	0.00000000000000000000000000000000000000000000000000000000000000E 0
3	-5.00000000000000000000000000000000000000000000000000000000000000E -1
4	-6.66666666666666666666666666666666666666666666666666666666666667E -1
5	-6.25000000000000000000000000000000000000000000000000000000000000E -1
6	-4.66666666666666666666666666666666666666666666666666666666666667E -1
7	-2.56944444444444444444444444444444444444444444444444444444444444E -1
8	-4.64761904761904761904761904761904761904761904761904761904761904E -2
9	1.53993055555555555555555555555555555555555555555555555555555556E -1
10	3.09744268077601410934744268077601410934744268077601410934744427E -1
11	4.18945932539682539682539682539682539682539682539682539682539682E -1
12	4.80134179092512425845759179092512425845759179092512425845759184E -1
13	4.96212223508230452674897119341563786008230452674897119341563794E -1
14	4.72883324237490904157570824237490904157570824237490904157570824E -1
15	4.17443394611983401665941343481031020713560396100078639761179444E -1
16	3.37869900654044172562691081209599728113246636765155283673802194E -1
17	2.42152881277598384493225763067032908302749572590842432112273384E -1
18	1.37821987671673031928245124634791612693884713803792546400700774E -1
19	3.16304777287616971764164591912975216914676404891458462063984322E -2
20	-7.06367673607733408609168269757651709262763040999481774622240174E -2
21	-1.64258811827792959953337607485686470366762236596590091074304514E -1
22	-2.45600815518869122900722274281531209340236351517363813987033344E -1
23	-3.12081781973129591036829352846586132481269081600040376131591164E -1
24	-3.62103498495764404426287021350265357551766950305207506177185994E -1
25	-3.94953331059299250406755327776696891679670451551609026380164944E -1
26	-4.10691037077920732531734688835003288775270994686090124719818144E -1
27	-4.10027637592329052554532740282023544106607247904395406787953214E -1
28	-3.94202599659302655074466688331681436495797445525563626305862794E -1
29	-3.64864077378427105537457756510572388567666655891491422758637644E -1
30	-3.23955703370649778207329899700171501307138077765784073721363794E -1
31	-2.73612428049616702514623641460111593578389182985407500527953604E -1
32	-2.16066917157274477569607145051662921789974472456481542581816214E -1
33	-1.53567612319153119730067347123113804690110193673194726153138 5E -1
34	-8.83026617997883751665721689159415152368946097242738449937566004E -2
35	-2.23717785957052685255159027467466014969906059611920933313552524E -2
36	4.23203873338527181674392101818069318931158721504070966644447794E -2
37	1.04039982339427093615244480801739341247577284976950556475238014E -1

38	1.61279533768169537466751109491356297883064678943940123953976 34E -1
39	2.12768690060034297599356218755158168837332701836064206797395903E -1
40	2.57481950543062671636270011915909782431189630654848011686404 74E -1
41	2.94640330760183769531354209949744861089325936753276065138792 0E -1
1	0.0000000000000000000000000000000000000000000000000000000000000000E 0
2	-2.5000000000000000000000000000000000000000000000000000000000000000E 0
3	-2.1666666666666666666666666666666666666666666666666666666666666667E 0
4	-7.9166666666666666666666666666666666666666666666666666666666666667E -1
5	7.5000000000000000000000000000000000000000000000000000000000000001E -2
6	5.6805555555555555555555555555555555555555555555555555555555555556E -1
7	7.9424603174603174603174603174603174603174603174603174603174603E -1
8	8.3712797619047619047619047619047619047619047619047619047619048E -1
9	7.6115795855379188712522045855379188712522045855379188712522046E -1
10	6.1517333553791887125220458553791887125220458553791887125220459E -1
11	4.3534278799037999037999037999037999037999037999037999037999037999004E -1
12	2.4765620198345892790337234781679226123670568115012559457003901E -1
13	7.0015436113526391304169081946859724637502415280193057970835748E -2
14	-8.6019683751268572697144125715554286982858411429840001268572693E -2
15	-2.1378225247969378260383551388842394133994244047154100064152 97E -1
16	-3.1030393691783174876759532844188928845013501098157182813267469E -1
17	-3.7533842631675615780716154198880622783517274647059894492200748E -1
18	-4.1057031653120782795125526045054350203570385470305932317897598E -1
19	-4.1898115858128306010597125548629217023150294964543730976879935E -1
20	-4.0434804458900102991891601828754077567833089534947666114362813E -1
21	-3.7085333771465043916955951398275003047212992486315157380551266E -1
22	-3.2278701830343241312329719266666570052846689242086592158209598E -1
23	-2.6432565312223191260905649242963381032603744379129397724743929E -1
24	-1.9937423206306494089549072199884490740993196873941234243872642E -1
25	-1.3145908766832598747537364241484869042190541433245126879120593E -1
26	-6.3661845115686130253225079990969583650440840733668998950420424E -2
27	1.4141308850882753094905879979901998525267528077401180061670805E -3
28	6.1651997633020024474342984274778007818414141173430482443270920E -2
29	1.1541045150365482388421177240296609072759995977268353661381457E -1
30	1.6149854061836327988996590368197073017793684066586347481518305E -1
31	1.9914044054158570366055186969896937450644107821538645337023162E -1
32	2.2793360694323888342228404898550636840654176892739055905168733E -1
33	2.4780307108794968518759627047271919544245802635029572550019935E -1
34	2.5895408908093393366058642051203860974510027613833507687251820E -1
35	2.6182488285708892453756282347411565032203089528080832094265998E -1
36	2.5704081006877347547577176264623438987457837281282359146479464E -1
37	2.4537096236938920144107228285803530873451669773167567709612483E -1
38	2.2768790915436402328342363726296819361569854017722453387882525 8E -1
39	2.0493106950056033815363064037797231217387468509459939923114942E -1
40	1.7807400248010760610151689361914017676412387022559342627234857E -1
0	-1.7590434063350465833680304369133133670653721225764939529432385 6E -2
1	3.8300252706956445180990777918504968737776688217985540489691359E -2
2	6.8492963778503455936638728306752217253688316709630195843444817E -3
3	-1.6272078625169271880508400874379120145541183637782360396839820E -1
4	7.9387753598921447779750422250706166938605469015016611152241913E -3
5	5.1753906521924039682864500675135987298106913772085529146769554E -2
6	7.5586549929810242092100387681532781092252304414831460671797425E -2
7	6.5075787785235327836431494681539386765079620796847040723267625E -2
8	6.2790903216663971403104955358352790106522081404166374879865 83E -2
9	5.6098196411004333803891426960964513834217048697014712304393425E -2
10	4.5843922500714942066580665982717236121455347409645472591423626E -2
11	3.6972333763694964919035776488072777567448729422918147314642743E -2

10	4.58439225007149420665806659897172361214505474096454725914236626E -2
11	3.63723337536389689190357764830727775674872049201681473464274236E -2
12	3.31199559973959098625428893036459022485875746323217858642866652E -2
13	2.63694381296580381530322627854971815777614321211580943599006606E -2
14	2.41557706192114920002160044393485201108569315905256986@2437769E -2
15	1.92714025723102148186277545581615335878691360567664062279101l4E -2
16	1.78095639309519127784437369428580092921049309963951797553149911E -2
17	1.41784523626108262293586127043747284764018895739078245715426455E -2
18	1.32301328930009416806995122095162411424659744200791349587126666E -2
19	1.04548715961986244148104163966115139333819454791149649330855766E -2
20	9.85761032627723804416547969550540283261092234842461673801362809E -3
21	7.68191411864092021061970584269882225697457895695366634069287596 -3
22	7.32937541543134862120422449902369567541593518106009048978306626 -3
23	5.58496774945220086039257938833629186489188791436217021746291416 -3
24	5.40776286304516946573465593570380827065580592099849874893498386 -3
25	3.98149079327277986383469222574915624836452637174935074029106986 -3
26	3.93405486550886379564423398495208544123054378397442706434758336 -3
27	2.74799137926677333927576702178013342206829559352149398187519146 -3
28	2.80026132920921635701838334789982216335912031109587573696686476 -3
29	1.79955707527552230667134400378146540329005896948241768682533226 -3
30	1.93178302410669667321275114217700869557970833816700221441623206 -3
31	1.07710900251938579174458163045970788803052647153106979250789756 -3
32	1.27662104777546915780700698003417546792656247476593722042909876 -3
33	5.39370040828607468212621731830578018043731678902529115367788286 -4
34	7.98564916756601266909592717275988865170357105706993942384742186 -4
35	1.57748894400572884535579541386227151333831611092365756610566056 -4
36	4.72854604166699247485519265814559151590645923703338790471079886 -4
37	-8.69256780856586498237276647005906492792213609256681953695139806 -5
38	2.83432136854674473113562952125012215966153069583320274115653066 -4
39	-2.06451732919776697913887043121594422584048556180082625898005096 -4
40	1.10631660775789093325050589253656205701491900991505630781910946 -4

APPENDIX C

Accounting information from actual execution of the runstream shown in Figure 1 on System I, and from actual execution of the runstream shown in Figure 2 on System I. The printed output of the latter execution appears in Appendix B.

```
RUNID: LOZI01   ACCT: 35650-LOZIER   PROJECT: CNSLT
TIME:     TOTAL: 00:00:10.264       CBS:  00000430.844
          CPU:   00:00:02.610       I/O:  00:00:02.735
          CC/ER: 00:00:04.918       WAIT: 00:00:00.000
IMAGES  READ:    0          PAGES:  11
TAPE  DRIVES:    0          MOUNTS: 0
START:     18:39:25 MAY 16,1980    FIN:  18:39:39 MAY 16,1980

**********U1108I****** APPROX. RUN COST **********************
TOTAL RUN COST: $   0.69
CPU:$   0.16      CBS:$   0.06      I/O:$   0.16    CC/ER:$   0.30
IMAGES  READ: $   0.00  PAGES: $   0.77
TAPES: $    0.00
PRIORITY: N          INPUT DEVICE: @@@@ST  (DISCOUNT APPLIED)
SURCHARGES: RUN: $0.25 PRINT:$0.25
(PRINT/PUNCH CHARGES ASSUME PRINT/PUNCH HAS OCCURRED)
(30% DISCOUNT ALL JOBS THRU 09/30/80)
```

```
RUNID: LOZ103    ACCT: 35650-LOZIER    PROJECT: CNSLT
TIME:     TOTAL: 00:01:18.649        CBS:   00003436.727
          CPU:   00:01:04.635        I/O:   00:00:05.692
          CC/ER: 00:00:08.321        WAIT:  00:00:00.000
IMAGES  READ:    0          PAGES:   18
TAPE  DRIVES:    0          MOUNTS:  0
START:     18:49:02 MAY 16,1980     FIN:   18:51:11 MAY 16,1980

**********U1108I****** APPROX. RUN COST **********************
TOTAL RUN COST: $    1.89
CPU:$    3.88       CBS:$    0.51       I/O:$    0.34     CC/ER:$    0.50
IMAGES  READ: $    0.00  PAGES: $    1.26
TAPES: $     0.00
PRIORITY: N           INPUT DEVICE: @@@@ST  (DISCOUNT APPLIED)
SURCHARGES: RUN: $0.25 PRINT:$0.25
(PRINT/PUNCH CHARGES ASSUME PRINT/PUNCH HAS OCCURRED)
(30% DISCOUNT ALL JOBS THRU 09/30/80)
```

NBS-114A (REV. 9-78)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBSIR 80-2138	2. Gov't. Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

Multiprecision Computing at NBS: Yesterday, Today, and Tomorrow

5. Publication Date: October 1980

6. Performing Organization Code

7. AUTHOR(S)

Daniel W. Lozier

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, DC 20234

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

13. Type of Report & Period Covered

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

Multiprecision computing is a technique by which arithmetic operations may be performed on a computer to precision levels that are higher than the directly supported single and double precisions. The last ten years have seen the development of portable Fortran software of very high quality that essentially duplicates all the capabilities of standard Fortran, so that an existing standard Fortran program can be re-executed to arbitrarily high precision. In this paper some of the design techniques for such software, which have evolved at NBS and elsewhere, will be discussed. Methods for using the software presently available at NBS will be described, and a complete example will be given. Directions for further extensions and improvements will be indicated.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)*

AUGMENT and MP; Fortran arithmetic extensions; multiple precision; multiprecision Fortran software; multiprecision mathematical functions; precompilers for special arithmetic.

18. AVAILABILITY ☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Sup. of Doc., U.S. Government Printing Office, Washington, DC 20402, SD Stock No. SN003-003-

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PRINTED PAGES
UNCLASSIFIED	52
20. SECURITY CLASS (THIS PAGE)	22. Price
UNCLASSIFIED	$7.00

USCOMM-DC